

TenBasket Technical Handover Guide

TENBASKET.COM | BUYER HANDOVER | AUGUST 2026

1. System overview

TenBasket is a responsive grocery-commerce web application. It has a public storefront, guest and account checkout, order history, rewards, contact and policy pages, and a protected administration area. The supplied application uses React 19 with TypeScript and Vite for the browser application, Convex for server functions, data, and product-image storage, Clerk for authentication, and Stripe Checkout for card payments. Cash on delivery is also supported.

The business currently reports \$0 monthly revenue. This technical guide explains the supplied application package; it does not represent traffic, customers, supplier operations, or financial performance.

2. Application areas and roles

- Public storefront: home, product categories, product detail views, basket, contact, delivery information, and return policy.
- Checkout: guest or signed-in purchase flow, delivery details, rewards redemption, card checkout, and cash on delivery.
- Customer area: account authentication, order history, rewards balance, member tiers, and referral link.
- Administration: product management, order management, manual orders, delivery configuration, feedback, reward administration, and optional customer announcements.
- Server layer: data queries and mutations, authentication checks, payment session creation and webhook handling, managed image storage, contact handling, and optional integrations.

The application has customer and administrator roles. It does not contain a separate supplier or multi-vendor role.

3. Buyer-safe architecture

The browser application is built into static assets that can be served by a standard web host. Browser requests use the buyer's Convex deployment for application data and server actions. Clerk provides user identity. The server validates administrative access before protected operations. Stripe Checkout handles card entry outside the storefront, and the payment webhook confirms successful or failed sessions. Product images use Convex managed storage.

Optional services include SMTP delivery for contact messages, WhatsApp customer announcements, and Google-powered voice product search. The core storefront can run without optional messaging, email delivery, or voice search, but those controls should be hidden or explained if the buyer leaves them unconfigured.

4. Relative project structure

- `App.tsx`: application routing, basket state, and shared customer interactions.
- `pages/`: storefront, checkout, orders, rewards, authentication, policies, contact, and administration screens.
- `components/`: layout, product, basket, feedback, and reusable interface components.
- `hooks/`: typed browser hooks connecting screens to server functions.
- `convex/`: schema, server queries, mutations, actions, payment handling, storage, and optional integrations.
- `public/`: static browser assets.
- `.env.example`: public configuration template with placeholders.

Generated build output and local environment files should not be treated as source of truth. Never place real secrets in browser-facing files.

5. Buyer-owned services and configuration

Use Node.js 22 or newer and pnpm. Create buyer-owned accounts for every retained service. The browser environment requires placeholder values such as:

```
VITE_CONVEX_URL=https://YOUR_DEPLOYMENT.convex.cloud
VITE_CLERK_PUBLISHABLE_KEY=YOUR_PUBLIC_AUTH_KEY
```

Configure server-only values in the Convex deployment environment, not in `.env.local` and not in built browser files:

```
CLERK_SIGNING_KEY=YOUR_SERVER_SIGNING_KEY
STRIPE_SECRET_KEY=YOUR_PAYMENT_SECRET_KEY
STRIPE_WEBHOOK_SECRET=YOUR_WEBHOOK_SIGNING_SECRET
SITE_URL=https://YOUR_DOMAIN
APP_ID=YOUR_APPLICATION_IDENTIFIER
ADMIN_CLERK_IDS=YOUR_ADMIN_USER_IDENTIFIERS
ADMIN_EMAILS=YOUR_ADMIN_EMAIL_ALLOWLIST
SEED_SECRET=YOUR_ONE_TIME_SEED_SECRET
```

Optional server configuration:

```
GOOGLE_API_KEY=YOUR_OPTIONAL_VOICE_SEARCH_KEY
SMTP_HOST=YOUR_MAIL_HOST
SMTP_PORT=YOUR_MAIL_PORT
SMTP_USER=YOUR_MAIL_USER
SMTP_PASSWORD=YOUR_MAIL_PASSWORD
CONTACT_RECIPIENT_EMAIL=YOUR_CONTACT_INBOX
WHATSAPP_TOKEN=YOUR_OPTIONAL_MESSAGING_TOKEN
WHATSAPP_PHONE_ID=YOUR_OPTIONAL_MESSAGING_PHONE_IDENTIFIER
```

Values beginning with `VITE_` are exposed to the browser and must only contain public configuration. Payment secrets, signing keys, mail passwords, seed secrets, and messaging tokens are server-only.

6. Installation and deployment

- 1 Install Node.js 22 or newer and enable pnpm.
- 2 Open the root of the supplied application package.
- 3 Install the locked dependencies:

```
pnpm install --frozen-lockfile
```

- 4 Create the buyer's development and production service environments.
- 5 Add only the public browser values to the local environment file or hosting build settings.
- 6 Add server-only values to the buyer's Convex environment.
- 7 Deploy server functions:

```
pnpm convex:deploy
```

- 8 Build the browser application once configuration is complete:

```
pnpm build
```

- 9 Publish the generated `dist/` contents to the chosen web host.
- 10 Point the domain to the host, enable HTTPS, and set `SITE_URL` to the final HTTPS origin.

For development, use `pnpm dev` and the buyer's non-production service accounts. Do not copy live secrets into a developer's browser environment.

7. Service setup

Authentication

Create a Clerk application for the buyer, copy its public browser key to the frontend environment, and set the server signing key in Convex. Configure the buyer's administrator user identifiers or approved administrator email addresses. Test a normal customer account and an administrator account separately. Remove all demonstration identities and seller access after handover.

Application data and image storage

Create a buyer-owned Convex deployment and deploy the schema and functions. Product records contain names, descriptions, prices, units, categories, visibility, stock state, featured state, and image references. Orders contain customer, delivery, payment, rewards, and fulfilment status data. Product uploads use Convex storage. Establish an export or backup routine before loading commercial data.

If sample data is required for a clean environment, set a temporary seed secret and run the idempotent seed function once:

```
pnpm exec convex run seed:ensure '{"secret":"YOUR_SEED_SECRET"}'
```

Remove or rotate the seed secret immediately afterward, and delete demonstration customers and orders before launch.

Card payments and cash on delivery

Create a buyer-owned Stripe account, configure the server secret key, and create a webhook endpoint that targets the application's payment webhook route. Subscribe to the checkout completion, checkout expiration, and asynchronous payment failure events used by the application. Store the webhook signing secret on the server. Begin in test mode and verify that successful checkout confirms the order, expired or failed checkout cancels it, and duplicate webhook delivery does not double-process the event.

Cash on delivery does not send the shopper to Stripe. It creates an unpaid order for the store owner. When the order is marked Delivered, the application records cash-on-delivery payment as paid. The buyer must define the real collection, receipt, cancellation, and refund procedures.

Contact email, messaging, and voice search

Contact submissions are saved even when SMTP is not configured. With SMTP settings present, a copy is sent to the chosen inbox. Test sender authentication and delivery before launch.

The optional WhatsApp controls require a buyer-owned approved business setup, template approval where applicable, valid customer consent, and the two messaging variables. The optional voice product search requires a Google API key. Disable or hide optional controls that the buyer does not intend to operate.

8. Security and credential rotation

- 1 Create new buyer-owned accounts and credentials instead of continuing with seller credentials.
- 2 Rotate authentication signing keys, payment keys, webhook secrets, mail passwords, messaging tokens, API keys, deployment tokens, and hosting credentials.
- 3 Replace administrator allowlists and remove demonstration users.
- 4 Confirm that server-only values are absent from browser bundles, screenshots, support messages, and source history.
- 5 Enforce HTTPS and least-privilege access for hosting, domain, data, payment, and authentication accounts.
- 6 Review customer-data retention, privacy notices, cookie use, deletion requests, and breach procedures for the buyer's jurisdiction.
- 7 Keep dependencies patched and review security advisories before each release.

9. Functional verification checklist

- Home page and category routes load on desktop and mobile.
- Visible, hidden, featured, in-stock, and out-of-stock products behave correctly.
- Basket quantities persist, update, and clear after a successful card checkout.
- Guest checkout validates phone, region, address, and delivery slot.
- Customer sign-up, sign-in, sign-out, and account order history work.
- Card checkout succeeds in test mode and the webhook confirms the matching order.
- Failed or expired card checkout does not leave a valid paid order.
- Cash-on-delivery orders appear as unpaid and can be moved through each status.
- Guest order lookup only returns orders for the same guest session and phone number.
- Rewards earning, redemption limits, cancellation restoration, tiers, and referrals match the published rules.
- Administrator access is blocked for ordinary customers.
- Product add, edit, image upload, visibility, stock, feature, and delete controls work.
- Order filters, details, manual order creation, status updates, and deletion work.
- Regions, delivery slots, payment instructions, feedback, and reward administration work.
- Contact delivery, optional messaging, and optional voice search are tested if enabled.
- Domain redirects, HTTPS, custom error handling, and policy links work in production.

10. Backup, monitoring, and maintenance

- Keep an encrypted backup of the transferred source package and each production release.
- Schedule Convex data exports or backups appropriate to order and customer data.
- Monitor payment webhooks, authentication activity, server logs, failed actions, and customer contact delivery.
- Add uptime and client-error monitoring for the public domain.
- Record domain, SSL, hosting, payment, authentication, and other renewal dates.
- Review dependencies, browser compatibility, mobile layouts, and checkout after material updates.
- Test data restoration and maintain a written incident and rollback procedure.

11. Boundaries and support

The supplied package does not include grocery inventory, supplier agreements, delivery operations, merchant approval, messaging approval, legal compliance, marketing, or guaranteed commercial results. Third-party services can change their pricing and requirements. The buyer is responsible for reviewing current vendor documentation and local regulations.

The seller provides up to 30 days of reasonable technical handover support for the supplied application and initial configuration questions. Ongoing development, business operations, paid services, legal work, and custom integrations are separate from the sale.